SECURITY AND COMMUNICATION NETWORKS
*Security Comm. Networks* 2011; **00**:1–13
DOI: 10.1002/sec

# Enabling Interoperation Between Shibboleth and Information Card Systems

Haitham S. Al-Sinani and Chris J. Mitchell

Information Security Group
Royal Holloway, University of London
[Haitham.Al-Sinani.2009, C.Mitchell]@rhul.ac.uk

## ABSTRACT

Whilst the growing number of identity management systems have the potential to reduce the threat of identity attacks, major deployment problems remain because of the lack of interoperability between such systems. In this paper we propose a scheme to provide interoperability between two widely discussed identity systems, namely Shibboleth and an Information Card system such as CardSpace or Higgins. When using this scheme, Information Card users are able to obtain an assertion token from a Shibboleth-enabled identity provider that can be processed by an Information Card-enabled relying party. The scheme is based on a browser extension and operates with both the CardSpace and the Higgins identity selectors without any modification. We specify the operation of the scheme and also describe an implementation of a proof-of-concept prototype. Additionally, security and operational analyses are provided. Copyright © 2011 John Wiley & Sons, Ltd.

**KEYWORDS**
Identity Management; CardSpace; Shibboleth; Interoperation; Browser Extension

Received . . .

## 1. INTRODUCTION

A number of identity management systems have been designed in order to simplify management of identities and mitigate identity-oriented attacks, e.g. CardSpace [1], Shibboleth [2], OpenID [3], and OAuth [4]. Most identity management architectures [5, 6, 7] involve the following main roles:

- the identity provider (IdP), which issues a security token to a user;
- the service provider, or relying party (RP) in CardSpace terminology, which consumes the security token issued by the IdP in order to identify the user, before granting him/her access; and
- the user, or principal in Liberty/Shibboleth terminology. The user employs a user agent (UA), i.e. software such as a web browser which sends requests to, and receives responses from, web servers. Typically, the UA processes protocol messages on behalf of the user, and prompts the user to make decisions, provide secrets, etc.

An IdP supplies a UA with a security token that can be consumed by a particular RP. Whilst one RP might solely support an Information Card system, another might only support Shibboleth. Therefore, to make these systems available to the largest possible group of users, effective interoperability between systems is needed. In this paper we investigate a case involving an Information Card-enabled RP, a Shibboleth-enabled IdP, and a UA that is only Information Card-enabled. The goal is to develop an approach to integration that is as transparent as possible to IdPs, RPs and identity selectors.

The scheme operates with a variety of Information Card-based systems, including CardSpace (described in section 2.1) and Higgins [8]. For simplicity of presentation, we restrict our description to the scheme's operation with CardSpace, a widely discussed example of an Information Card-based system.

We consider CardSpace-Shibboleth interoperation because of the latter's wide adoption (notably by educational institutions — see section 2.2.1). Complementing this, the wide use of Windows, recent versions of which incorporate CardSpace, means that enabling interoperation between the two systems is likely to be of significance for



*Prepared using secauth.cls [Version: 2010/06/28 v2.00]*

large numbers of identity management users and RPs. CardSpace-Shibboleth interoperation is also attractive since both schemes support user authentication as well as exchange of user attributes. In addition, they both support SAML tokens.

The remainder of the paper is organised as follows. Section 2 presents an overview of CardSpace and Shibboleth, and section 3 describes the proposed integration scheme. In section 4, we outline certain advantages of the scheme. Section 5 considers possible extensions, and, in section 6, we discuss implementation issues. In section 7 we describe a prototype realisation, and section 8 highlights possible areas for related work. Finally, section 9 concludes the paper. Some of the contents of this paper were presented at ACNS '11, Malaga, Spain, June 2011.

## 2. CARDSPACE AND SHIBBOLETH

### 2.1. CardSpace

#### 2.1.1. Introduction

CardSpace provides a secure and consistent way for users to control and manage personal information, to review personal data before sending it to a website, and to verify the identity of visited websites. It also enables websites to obtain data from users, e.g. to support user authentication and authorisation.

Digital identities are represented to users as Information Cards (or InfoCards), XML-based files that list the types of claim made by one party about itself or another party. The concept is inspired by real-world cards, such as driving licences and credit cards. A user can employ one InfoCard with multiple websites, or can use separate InfoCards at different websites, helping to enhance user privacy and security. There are two types of InfoCards: personal (self-issued) cards, and managed cards issued by remote IdPs. Personal cards are created by users themselves, and the claims listed in such an InfoCard are asserted by the self-issued identity provider (SIIP) that co-exists with the CardSpace identity selector (or just the selector) on the user machine. InfoCards, personal or managed, do not contain sensitive information, but instead carry metadata indicating the types of personal data associated with this identity, and from where assertions regarding this data can be obtained. The data referred to by personal cards is stored on the user machine, whereas the data referred to by a managed card is held by the IdP that issued it [1, 9, 10].

By default, CardSpace is supported by Internet Explorer from version 7 onwards. Extensions to other browsers, such as Firefox and Safari, also exist. An updated version, CardSpace 2.0 Beta 2, was released, although Microsoft announced in early 2011 that it will not ship; instead Microsoft has released a technology preview of U-Prove [11, 12]. In this paper, unless explicitly stated, we refer throughout to the CardSpace version that is shipped by default as part of Windows Vista and Windows 7, that is available as a free download for XP and Server 2003, and which has been approved as an OASIS standard [13].

#### 2.1.2. Operational Protocol

In order to maximise interoperability with non-Windows platforms, CardSpace has been specifically designed to use open standards-based protocols, notably the WS-* standards including WS-Policy/WS-Security Policy [14, 15], WS-MetadataExchange [16], WS-Trust [17] and WS-Security [18]. Note that HTML/XHTML and/or HTTP/S can be used in place of most of these standards; e.g. instead of using WS-Policy/WS-SecurityPolicy, a website can simply describe its policy in HTML/XHTML.

The integration scheme makes use of CardSpace personal cards, and so we next describe their operation. Note that the scheme does not affect the use of managed cards.

The selector allows a user to create a personal card and populate its fields with self-asserted claims. CardSpace restricts the contents of personal cards to non-sensitive data in the form of 14 editable claim types, namely *First Name, Last Name, Email Address, Street, City, State, Postal Code, Country/Region, Home Phone, Other Phone, Mobile Phone, Date of Birth, Gender* and *Web Page*. Data inserted in personal cards is stored in encrypted form by the SIIP on the user machine.

When using personal cards, CardSpace adopts the following protocol. We describe the protocol for the case where the RP does not employ a security token service (STS), software responsible for security policy and token management within an IdP and, optionally, within an RP [19, 20].

1. UA → RP: HTTP/S Request. The user employs a UA to navigate to a CardSpace-protected web page.
2. RP → UA: HTTP/S Response. A login page is returned containing the CardSpace-enabling tags in which the RP security policy is embedded.
3. User → UA. The RP page offers the option to use CardSpace; selecting this option activates the selector, which is passed the RP policy. If this is the first time that this RP has been contacted, the selector will display the identity of the RP, giving the user the option to either proceed or abort the protocol.
4. Selector → InfoCards. The selector, after evaluating the RP policy, highlights those


*Security Comm. Networks* 2011; **00**:1–13 © 2011 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
*Prepared using secauth.cls*

InfoCards matching the policy and greys out the rest. InfoCards previously used for this RP are displayed in the upper half of the selector screen.

5. User → Selector. The user chooses a personal card. Alternatively, the user could create and choose a new personal card. The user can preview the card (with its associated claims) to ensure that they are willing to release the claim values. Of the claims specified in an InfoCard, only those requested in the RP policy will be passed to the requesting RP.
6. Selector ⇌ SIIP. The selector creates and sends an XML-based Request Security Token (RST) to the SIIP, which responds with an XML-based Request Security Token Response (RSTR).
7. Selector → UA → RP. The RSTR (i.e. the SIIP-generated SAML token) is passed to the UA, which forwards it to the RP.
8. RP → UA. The RP verifies the token, and, if satisfied, grants access.

#### 2.1.3. Private Personal Identifiers (PPIDs)

When a user creates a new personal card, CardSpace generates an ID and a master key for this card. The card ID is a globally unique identifier (GUID), and the master key is 32 bytes of random data. When a user first uses a personal card at a particular RP, CardSpace generates a site-specific PPID by combining the card ID with data taken from the RP certificate, and a site-specific signature key pair by combining the card master key with data taken from the RP certificate. The RP domain name and/or IP address is used if no RP certificate is available.

Since the PPID and key pair are RP-specific, the PPID does not function as a global user identifier, helping to enhance user privacy and reduce the impact of PPID compromise. The selector displays a shortened version of the PPID to protect against social engineering attacks and improve readability.

When a user first interacts with an RP using CardSpace, the RP retrieves the PPID and the public key from the received SAML token, and stores them. If a personal InfoCard is re-used at a site, the supplied security token will contain the same PPID and public key as used previously, and will be signed using the corresponding private key. The RP compares the received PPID and public key with its stored values, and verifies the digital signature.

The PPID could be used on its own as a shared secret to authenticate a user to an RP. However, it is recommended that the associated (public) signature verification key, as held by the RP, should always be used to verify the signed security token to provide a more robust authentication method [1].

#### 2.1.4. Proof of Ownership

In many identity management systems it is important for the user to have the means to prove to an RP that it owns the assertion generated by an IdP; such mechanisms are often referred to as proof-of-possession methods.

A SAML token can be coupled with cryptographic evidence to demonstrate the sender's rightful possession of the token [19, 20]. To achieve this, a security token can be associated with symmetric or asymmetric proof keys. If a symmetric key token is requested by the RP, a shared secret proof key is established between the selector and the CardSpace-enabled IdP [21], which is then revealed to the RP. If an asymmetric key token is requested, the selector generates an ephemeral RSA key pair and sends the public part of the key (along with a supporting signature proving ownership of the corresponding private key) to the CardSpace-enabled IdP [21]. If approved by the IdP, the public part is then sent to the RP in the SAML token and the private part of the RSA key pair is used by the client to prove the subject's rightful possession of the security token. Although the use of such a key may not be as efficient as the symmetric approach, it helps to protect user privacy since the RP identity does not need to be disclosed to the IdP.

Note that the default behaviour of the selector is different in the special case of browser-based client interactions with a website, in which case 'bearer' tokens are requested. Because a web browser is only capable of submitting a token to a website passively over HTTP without any proof-of-possession, bearer tokens with no proof keys are used [19, 20].

### 2.2. Shibboleth

#### 2.2.1. Introduction

The Shibboleth specifications define a set of interactions between an IdP and an RP to support single sign-on and attribute exchange [2]. It is estimated that over 4 million university students, staff, and faculty are involved in Shibboleth federations (see `http://en.wikipedia.org/wiki/Shibboleth_(Internet2)`). In addition to IdPs and RPs, the Shibboleth architecture includes an optional component called WAYF (Where Are You From), supporting IdP discovery. Alternatively, the role of the this component can be taken by the RP. Shibboleth supports the following profiles.

- **Browser Post Profile**. In this profile the SAML messages exchanged between the IdP and RP are embedded in HTML forms, which can be sent automatically by JavaScript-enabled browsers. The scheme proposed here supports this profile.
- **Artifact Profile.** This profile involves embedding an artifact (i.e. an opaque

reference) in a URL exchanged between the IdP and RP via browser redirection. It also requires direct (back-channel) RP-IdP communication, where the RP uses the artifact to retrieve the full SAML assertion from the IdP. As it requires direct RP-IdP communication, which is inconsistent with the CardSpace approach (in which RP-IdP interactions pass via the selector on the user machine), the integration scheme does not support this profile.

#### 2.2.2. Protocol Operation

We next describe the Shibboleth protocol, covering the main differences between the two profiles introduced above.

1. UA → RP. The user employs a UA to navigate to a Shibboleth-protected web page.
2. RP → UA. The RP generates an authentication request and redirects the UA to either a WAYF service or directly to an IdP. A WAYF service is typically used if the RP wishes to delegate the task of IdP discovery.
3. UA ⇌ WAYF (optional). If a WAYF service is used, it interacts (via unspecified means) with the UA to allow the user to select an IdP. The WAYF service then redirects the UA to the user-selected IdP with the RP's authentication request. Note that the WAYF component can offer the user the option to store their choice of IdP for subsequent logins.
4. IdP ⇌ User. If necessary, the IdP authenticates the user by some means outside the scope of Shibboleth. (Authentication may be unnecessary if a valid authentication session already exists.)
5. IdP→UA→RP. The IdP generates a digitally-signed SAML assertion (if the browser post profile is used) or a SAML artifact (if the artifact profile is used) and redirects the UA to the RP. Note that the SAML assertion may assert attributes in addition to asserting that the user has been authenticated. Note also that, if the browser post profile is used, the next step is skipped.
6. RP⇌IdP (optional). The RP uses the artifact received in the previous step to issue an attribute query to the IdP, which directly responds with a SAML response message. Note that this communication takes place via a mutually-authenticated back-channel.
7. RP→ UA. The RP verifies the token, and, if satisfied, grants access.

Two 'major' Shibboleth versions have been released, namely Shibboleth 1.3, which builds on the SAML 1.1 specifications [2, 22, 23], and Shibboleth 2.0 (`http://shibboleth.internet2.edu/shib-v2.0.html#new`), which builds on the SAML 2.0 standards; v2.0 is backward compatible with v1.3.

#### 2.2.3. Attributes

Shibboleth uses the SAML attribute request protocol to allow attribute sharing between IdPs and RPs. Such an attribute exchange is, however, optional since an RP may choose to request only an authentication assertion. Approximately 40 attributes have been defined as 'common' identity attributes, including the six 'highly recommended' attributes, namely *givenName, sn (surname), cn (common name), eduPersonScopedAffiliation, eduPersonPrincipalName* and *eduPersonTargetedID* [23].

#### 2.2.4. Proof of Ownership

As stated earlier, Shibboleth 2.0 builds on SAML 2.0, which offers three proof-of-possession methods (also referred to as subject confirmation methods): Holder-of-Key (HoK), Sender-Vouches, and bearer [24]. The HoK method [25] can be used to address both the symmetric and asymmetric proof-of-possession requirements of a CardSpace-enabled RP.

### 2.3. Comparison

Table I compares the CardSpace and Shibboleth systems [10, 26].

## 3. THE INTEGRATION SCHEME

The parties involved are a CardSpace-enabled RP, a CardSpace-enabled UA (e.g. a suitable web browser such as Internet Explorer), a Shibboleth-enabled IdP, and a browser extension implementing the protocol described below.

### 3.1. Preconditions

The scheme has the following requirements.

- The user must have an existing relationship with both the RP and the IdP (thus the IdP will have a means of authenticating the user). Note that both the RP and the user must trust the IdP.
- The RP must not employ an STS. Instead, the RP must express its security policy using HTML/XHTML, and interactions between the selector and the RP must be based on HTTP/S via a web browser (a simpler and probably more common scenario for RP interactions). This is because the scheme uses a browser extension, and is thus incapable of


*Security Comm. Networks* 2011; **00**:1–13 © 2011 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
*Prepared using secauth.cls*

managing the necessary communications with an RP-STS.

- To enable IdP-discovery, the browser extension must be able to operate a WAYF-like component and offer the user the option to store their choice of IdP for future logins.
- The RP must incorporate three pieces of functionality outside of that normally required of a CardSpace-enabled RP:
  - it must be capable of processing SAML 2.0 tokens;
  - it must be willing to accept, and be capable of processing, a 'CardSpace-like' SAML token provided by the browser extension, which includes a signed SAML assertion generated by an IdP and a signed SAML assertion generated by the SIIP on the client machine; and
  - it must be capable of verifying the IdP's signature on a SAML assertion, as included in the SAML token provided by the extension.
- The IdP must be prepared to provide SAML assertions for RPs for which a federation agreement does not exist for the user concerned. It is thus not necessary for the user to Shibboleth-federate the IdP with the RP (which would be difficult to achieve given that we are not requiring the RP to be Shibboleth-enabled).

### 3.2. Operation

The protocol operates as follows, with step numbers as shown in Fig. 1. Steps 1, 2, and 4–7 of the integration scheme are the same as steps 1, 2, and 3–6, respectively, of the CardSpace personal card protocol given in section 2.1.2, and hence are not described again here.

3. Browser Extension → UA. The extension performs the following steps.
   (a) It scans the login page to detect whether the RP website supports CardSpace. If so, it proceeds; otherwise it terminates.
   (b) It examines the RP policy to check whether the use of personal cards is acceptable. If so, it proceeds; otherwise it terminates, giving CardSpace the opportunity to operate normally.
   (c) It temporarily keeps a local copy of any RP-requested claims.
8. Selector → UA → IdP. Unlike in the 'standard' case, the RSTR is not sent to the RP; instead the browser extension intercepts the RSTR and performs the following steps.
   (a) It asks the user whether the use of Shibboleth-based authentication is required. If so, it proceeds; otherwise it terminates, giving CardSpace the opportunity to operate normally. Note that the browser extension could offer the user the option to store their answer for subsequent logins at this RP.
   (b) It displays a WAYF-like component to allow the user to select the appropriate IdP. Note that the browser extension could offer the user the option to store their selection for subsequent logins at this RP.
   (c) It constructs a SAML authentication request, and forwards it to the user-selected IdP. Note that this request will also indicate the RP-requested user attributes (if any) which are to be asserted by the IdP. The browser extension will know what they are since they were stored by it in step 3c.
9. IdP ⇌ User. If necessary, the IdP authenticates the user. If successful, the IdP generates and returns a digitally-signed SAML token to the UA, containing an authentication statement and, possibly, an attribute statement.
10. Browser Extension → UA → RP. The extension generates an unsigned SAML token (with a nonce and time-stamp) that contains both the (digitally-signed) SIIP-issued RSTR as well as the (digitally-signed) Shibboleth-issued token. The UA then forwards the browser extension-generated SAML token to the RP, optionally after first obtaining permission from the user.
11. RP → User. The RP verifies the received SAML token (including verifying the RSTR signature, PPID, the Shibboleth-enabled IdP's signature, nonces, time-stamps, etc.), and, if satisfied, grants access.

If an attribute assertion is required, the RP could, in step 11, compare the (locally-stored) SIIP-asserted attributes with the (remotely-stored) IdP-asserted attributes. Such a procedure potentially gives the RP added guarantees about the validity of these attributes.

Given that we have assumed that the RP supports SAML 2.0 tokens, the RP can therefore use the Shibboleth/SAML 2.0-supported HoK [25] method (which can be symmetric or asymmetric) to express its proof-of-possession requirements. However, a symmetric proof key should only be used if the user is willing to disclose the identity of the RP to the IdP, and if the RP holds a valid certificate. For

browser-based applications (and also where no proof-of-possession is needed), the scheme supports bearer tokens [19, 21, 24].

Note that the additional steps above can be integrated into the CardSpace framework relatively easily, as the prototype implementation shows.

# 4. ADVANTAGES OF THE SCHEME

We next describe possible advantages of the scheme.

## 4.1. Defeating Phishing

The scheme mitigates the risk of phishing. This is because the user interacts with a browser extension-operated WAYF running on the user machine to select the IdP. Hence, the RP will not be able to redirect the user to an IdP of its choosing. By contrast, in OpenID, Liberty and in some Shibboleth scenarios (e.g. where a WAYF component is not used), a malicious RP could redirect a user to a fake IdP, which might capture the user credentials [27].

## 4.2. Integration at the Client Side

IdPs and RPs may not wish to accept the burden of supporting two identity management systems simultaneously, unless there is a significant financial incentive. Currently, major Internet players do not support interoperation between identity management systems. As a result, a client-side technique for supporting interoperation could be practically useful. Such a technique could avoid the impact on the performance of the server (since the overhead is handled by the client), and could also reduce the load on the network.

# 5. POSSIBLE EXTENSIONS

## 5.1. Scope

The scheme proposed here applies to users of Information Card-enabled RPs (such as CardSpace). Since CardSpace users are currently only capable of retrieving security tokens from CardSpace-enabled IdPs, the scheme extends this capability to enable such users to obtain security tokens from non-Information Card-enabled IdPs, such as Shibboleth-enabled IdPs.

Interoperation between a CardSpace-enabled IdP and a Shibboleth-enabled RP is not supported. Indeed, without technical co-operation from the bodies responsible for developing the Shibboleth and CardSpace specifications, it appears likely to be difficult to implement bidirectional interoperation.

## 5.2. SAML-Compliant IdPs

Although the scheme is presented as Shibboleth-specific, we suspect that a modified version of the scheme could also be applied to other SAML-compliant IdPs. Given that SAML 2.0 builds on SAML 1.1, Liberty ID-FF 1.2 and Shibboleth 1.3, a mapping may be possible.

Modifying the scheme to interoperate with any SAML-aware IdP would clearly increase its applicability; this remains possible future work. Note that the scheme as defined here operates with a variety of Information Card-based systems, including CardSpace and Higgins.

## 5.3. U-Prove Tokens

U-Prove [11, 12, 28] is an anonymous credential system, support for which has been incorporated in CardSpace 2.0 [29].

At the heart of the U-Prove technology is the notion of a *U-Prove token*, a cryptographically protected container of user attribute information of any type. Given that CardSpace supports tokens of any type, the scheme presented here can be extended to support U-Prove tokens. Indeed, the user experience is precisely the same when using a card supporting U-Prove [28].

The protocol specified in section 3.2 can be extended to support the transfer of U-Prove tokens. In addition to supporting a SAML-based RSTR, the protocol can also be configured to support a U-Prove-based RSTR, which is also XML-encoded.

Note that, in CardSpace version 2.0, U-Prove tokens can only be chosen by selecting managed cards (as opposed to personal cards). This means that U-Prove tokens cannot be supported by the current scheme since it does not cover the case where the RP policy specifies use of a managed card (see section 2.1.2). However, we believe that the scheme could relatively easily be modified to support RP policies which request U-Prove tokens. (The U-Prove token can be identified by the URI: `http://schemas.xmlsoap.org/ws/2010/03/uprove/token`.)

# 6. IMPLEMENTATION ISSUES

## 6.1. Token Storage and Forwarding

The means by which the security token is forwarded to the RP and how/where the RSTR token is stored must be chosen carefully. We refer to the numbered protocol steps given in section 3.2.

The responsibility for delivering the security token could be given to the IdP (as is normally the case when using the browser post profile). In this case the RP address could be added to the SAML authentication request (as prepared in step 8) so



*Security Comm. Networks* 2011; **00**:1–13 © 2011 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
*Prepared using secauth.cls*

that the IdP knows to which RP it must forward the token (again as is normally the case for the Shibboleth profiles). Although this would avoid the need for changes to the normal operation of the IdP and would potentially also help auditing, such an approach has privacy implications since the IdP would learn the identity of the RP.

As a result, as specified in step 10 of the proposed scheme, the responsibility for sending the security token to the RP is given to the UA. Thus a means is required for giving the browser extension the address of the RP, so that it can forward the token. We next consider three possible ways in which the RP address might be made available.

- The RP address could be stored in the browser extension itself. Whilst this puts the user in control, it is not user-friendly, as it would require users to manually add the address of each RP into the code of the browser extension.
- After the security token is returned from the IdP, the browser extension could ask the user to enter the RP address, e.g. using a JavaScript pop-up box or an HTML form. However, this approach is inconvenient, since again it would require users to manually enter the the address of each RP.
- The browser extension could store the RP address as well as the RSTR message in encrypted form in one or more cookies as part of step 3, so that the browser extension is able to obtain them in step 10. In order to adhere to cookie security rules [30], this must be done in such a way that the browser believes it is communicating with the same domain when the cookie is set and when it is retrieved. Note that creation of and access to the cookie is handled by the browser extension transparently to RPs and IdPs.

  To achieve this, the browser extension encrypts and stores the RP address in a cookie in step 3, before the selector is invoked. As part of step 8, the browser extension retrieves the encrypted value from the cookie and sends it to the IdP as a hidden variable in an HTML form or as a URL parameter. Similarly, the extension in step 8 also encrypts the RSTR (after intercepting it) and sends it to the IdP as a hidden variable in an HTML form.

  As part of step 10, the IdP returns the encrypted RP address and the encrypted RSTR to the UA (again as a hidden form variable or as a URL parameter). The browser extension then retrieves the encrypted values and decrypts them to obtain the RP address and the RSTR.

  Note that the IdP is unable to read the RP address or the RSTR (thus protecting user privacy) since they are encrypted using a key known only to the browser extension. If the IdP, however, needs the RP address for auditing purposes (e.g. for legal reasons), or the IdP policy requires the disclosure of the RP identity (e.g. so that it can encrypt the security token using the RP's public key), then the RP address could be sent in plaintext to the IdP.

  Finally note that the Shibboleth specification allows the RP to use a hidden form variable called 'RelayState' to maintain state in an RP-UA-IdP session. A Shibboleth-compliant RP could insert data into this variable and the Shibboleth-enabled IdP must return this data intact in the same hidden form variable. We propose to make use of this 'RelayState' variable in our scheme to transfer/maintain the encrypted versions of the RP address and the RSTR message.

### 6.2. Attribute Handling

As stated in sections 2.1.2 and 2.2.3, CardSpace personal cards only support fourteen editable attributes, whereas Shibboleth supports many more. As the two systems use two different sets of attribute types, this clearly causes a problem in creating a SAML (attribute) request message for a Shibboleth IdP from a policy statement provided by a CardSpace-enabled RP. We outline two approaches to dealing with this problem.

1. We could restrict the RP to requesting only CardSpace personal card attributes. The browser extension would then need to convert the requested attributes to Shibboleth attributes, and include the converted attribute types in the SAML request message sent to the IdP. An example mapping is shown in Table II. Note that in this case the RP will still have to understand the Shibboleth attributes, since the IdP SAML assertion will include attributes in Shibboleth syntax. However, the browser extension-produced SAML token, generated in step 10, could be extended to include a means of CardSpace-Shibboleth attribute mapping.
2. Alternatively, the RP could be permitted to request any of the Shibboleth-supported attributes. If an attribute not supported by CardSpace personal cards is requested (and given that the RP permits the use of any IdP), then the browser extension would need to be configured to request this attribute from the user-selected IdP. However if any attributes are required that are outside the set permitted

in a personal card, then the selector will clearly not highlight any of the personal cards.
In order to cause the selector to highlight personal cards, the browser extension must modify the RP policy. In particular, as part of step 3 the browser extension must (after storing them) strip out the attributes that are outside the set supported by personal cards, and then request them from an IdP as part of step 8. Note, however, that such a modification will prevent CardSpace from operating normally in the case where a personal card is requested. Nevertheless, if the RP specifies the use of managed cards (i.e. does not permit personal cards), then CardSpace would still operate normally, since the extension will shut down if it sees such a policy statement.

To support the broadest range of user attributes, the browser extension could be configured to support both of the approaches described above.

## 7. PROTOTYPE REALISATION

We next give details of a prototype implementation of the scheme, which operates with the Shibboleth browser post profile.

### 7.1. Implementation Details

The prototype is coded in JavaScript [31], chosen because its wide adoption should simplify the task of porting the prototype to a range of other browsers. It uses the Document Object Model (DOM) to inspect and manipulate HTML pages and XML documents. The JavaScript code is executed using a C#-driven browser helper object (BHO), a Dynamic-link library (DLL) module designed as a plug-in for Internet Explorer. Once installed, the BHO attaches itself to Internet Explorer, thus gaining access to the current page's DOM. The prototype can readily be enabled or disabled using the add-on manager in the Internet Explorer Tools menu. Note that the integration plug-in does not require any changes to default Internet Explorer security settings, thus avoiding potential vulnerabilities that might result from such changes. Note also that the scheme operates with both the CardSpace and the Higgins (`http://wiki.eclipse.org/GTK_Selector_1.1-Win`) identity selectors without any modification.

### 7.2. Prototype Operation

We next consider specific operational aspects of the prototype. Prior to use, the user must have accounts with a CardSpace-enabled RP and a Shibboleth-enabled IdP. We refer throughout to the numbered protocol steps given in section 3.2.

In step 3 the plug-in uses the DOM to perform the following processes.

3.1 It scans the web page in the following way. (Note that the CardSpace documentation [19, 20] specifies two HTML extension formats that can be used to invoke the selector from a web page, both of which involve placing the CardSpace object tag inside an HTML form. This motivates the choice of web page search method.)

(a) It searches through the HTML elements of the web page to detect whether any HTML forms are present. If so, it searches each form, scanning through each of its child elements for an HTML object tag.

(b) If an object tag is found, it retrieves it and examines its type. If it is of type 'application/x-informationCard' (which indicates support for CardSpace), it continues; otherwise it aborts.

(c) It searches through the param tags (child elements of the retrieved CardSpace object tag) for the 'issuer' tag and examines its value; if it is '`http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self`', indicating that the use of personal (self-issued) cards is acceptable, it continues; otherwise it terminates. Note that the plug-in also continues if the value of the 'issuer' tag is set to 'any' or '$*$', or if the 'issuer' tag is absent, since the use of personal cards is acceptable in these cases.

(d) It retrieves the 'requiredClaims' and 'OptionalClaims' tags from the param tags. It obtains and temporarily stores in a cookie the mandatory and optional claim types listed in these tags.

3.2 It adds a JavaScript function to the head section of the HTML page to intercept the XML-based RSTR message before it is sent back to the RP (such a token will be sent by the selector in step 8).

3.3 It obtains the current action attribute of the CardSpace HTML form, encrypts it using AES [32] with a secret key known only to the plug-in, and then stores it in a cookie. This attribute specifies the URL of a web page at the RP to which the RSTR must be forwarded


*Security Comm. Networks* 2011; **00**:1–13 © 2011 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
*Prepared using secauth.cls*

for processing. If the attribute is not a fully qualified domain name, the JavaScript inherent properties, e.g. *document.location.protocol* and/or *document.location.host*, are used to help reconstruct the full URL.

3.4 It changes the current action attribute of the CardSpace HTML form to point to the newly created 'interception' function (see step 3.2 above).

3.5 It creates and appends an 'invisible' HTML form to the HTML page to be used later for sending the SAML token request to the IdP.

In step 8 the plug-in uses the DOM to perform the following steps.

8.1 It intercepts the RSTR message sent by the selector using the added function.

8.2 Using a JavaScript pop-up box, it asks the user whether the use of Shibboleth-based authentication is required. If so, it proceeds; otherwise it terminates, giving CardSpace the opportunity to operate normally. On proceeding, the plug-in offers to store the user's answer using a plug-in-embedded checkbox; if checked, the plug-in stores the user answer in a persistent cookie.

8.3 It encrypts the RSTR using AES with a secret key known only to the plug-in.

8.4 It prompts the user to select an IdP using a WAYF-like component, implemented as a plug-in-embedded HTML form containing a drop-down list.

8.5 It offers to store the user's choice of IdP using a plug-in-embedded checkbox; if checked, the plug-in stores the user selection in a persistent cookie.

8.6 It constructs a SAML request conforming to either SAML 1.1 syntax if the IdP is Shibboleth 1.3-complaint, or to SAML 2.0 syntax if the IdP is Shibboleth 2.0-compliant. The plug-in learns the IdP's version from the plug-in-operated WAYF. Note that this request will also indicate the RP-requested user attributes (if any) that are to be asserted by the IdP. The plug-in will know what they are since they were stored by it earlier (see step 3.1.d).

8.7 It writes the entire (Base64 encoded) SAML request message as a hidden variable (SAMLRequest) into the invisible HTML form created earlier (see step 3.5 above).

8.8 It retrieves the encrypted RP URL from the appropriate cookie and inserts it together with the encrypted version of the RSTR message into the invisible form as the hidden variable 'RelayState'.

8.9 It writes the URL of the IdP into the action attribute of the form.

8.10 It auto-submits the form (transparently to the user), using the JavaScript method 'click()' on the 'submit' tag, thus redirecting the user to the IdP.

In step 10, the plug-in operates as follows.

10.1 It recovers the encrypted string from the RelayState hidden variable and decrypts it using its internally stored secret key. The SIIP-issued RSTR and the RP URL are then recovered from the decrypted data.

10.2 It generates a SAML token with a unique ID, nonce and a time-stamp, referred to here as the 'user token'. The plug-in embeds the signed SIIP-issued RSTR (retrieved in the previous step) and the signed Shibboleth-issued SAML response message (after retrieving it from the SAMLResponse HTML hidden variable) into the (unsigned) user token.

10.3 It inserts the RP URL (retrieved in step 10.1) into the action attribute of the HTML form carrying the received SAML token.

10.4 It displays a representation of the token to the user and requests consent to proceed. The displayed text indicates the types of attributes the user token is carrying, as well as the exact RP URL to which the token will be forwarded. The JavaScript 'confirm()' pop-up box is used to achieve this.

10.5 If the user agrees to submission of the token, it seamlessly submits the token to the RP using the JavaScript 'click()' method.

The prototype has been successfully tested with experimentally-implemented websites (acting as the Shibboleth-enabled IdP and the CardSpace-enabled RP) as well as with the current (unmodified) CardSpace and Higgins (`http://wiki.eclipse.org/GTK_Selector_1.1-Win`) identity selectors.

### 7.3. Limitations

The plug-in must scan every browser-rendered web page to detect whether it supports CardSpace, and this may affect system performance. However, informal tests on the prototype suggest that this is not a serious issue. In addition, the plug-in can be configured so that it only operates with certain websites.

If the web browser is compromised, then an adversary could steal the user token (see above), and could use this to impersonate the user. Moreover, if the RP does not use HTTPS, then the SIIP-issued RSTR will not be encrypted. Assuming that the web browser is not a secure environment, it may be possible for a malicious plug-in or other malware to get access to sensitive information disclosed by the plaintext RSTR and/or the user token. However, the same risks apply when manually entering credentials (e.g. username-password) into the browser [33].

Finally note that some older browsers (or browsers with scripting disabled) may not be able to run the integration plug-in, as it was built using JavaScript. However, most modern browsers support JavaScript (or ECMAscript), and hence building the prototype in JavaScript is not a major usability obstacle.

## 8. RELATED WORK

A somewhat similar scheme [34] has previously been proposed to support CardSpace-Liberty interoperation. However, unlike the scheme proposed here, the CardSpace-Liberty integration scheme does not support the exchange of identity attributes and does not operate with HTTPS-enabled websites. Two further similar schemes have recently been proposed, allowing client-based interoperation between a CardSpace-enabled RP and an OpenID-enabled IdP [35] or an OAuth-enabled IdP [36].

Another scheme supporting interoperation between CardSpace and Liberty has been proposed by Jørstad et al. [37]. In this scheme, the IdP is responsible for supporting interoperation. The IdP must therefore perform the potentially onerous task of maintaining two different identity management schemes. This scheme also requires the user to possess a mobile phone supporting the Short Message Service (SMS). Moreover, the IdP must always perform the same user authentication technique, regardless of the identity management system the user is attempting to use. The IdP simply sends an SMS to the user, and, in order to be authenticated, the user must confirm receipt of the SMS. This confirmation also serves as an implicit indication of user approval for the IdP to send a security token to the RP. By contrast, the scheme proposed in this paper supports interoperation between CardSpace and Shibboleth, does not require use of a handheld device, and does not enforce a specific authentication method.

Finally, in 2007, Internet2 announced plans to develop extensions to Shibboleth to support CardSpace (`https://lists.internet2.edu/sympa/arc/i2-news/2007-05/msg00009.html`). This included collaboration with Microsoft in order to add Information Card support to Shibboleth. However, unlike the integration scheme proposed in this paper, this approach does not seem to be based on a browser extension running on the client machine. Instead, interoperation appears to be provided by Shibboleth IdPs/RPs (`https://lists.internet2.edu/sympa/arc/shibboleth-dev/2007-05/msg00021.html`), which is likely to necessitate significant changes to the servers.

## 9. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a means of interoperation between two leading identity systems, namely CardSpace and Shibboleth. CardSpace users (indeed, users of any Information Card system) are able to obtain an assertion token from a Shibboleth-enabled identity provider that can be processed by a CardSpace-enabled relying party. The scheme uses a browser extension, requires no major changes to identity providers and relying parties, and does not require any changes to the deployed CardSpace identity selector.

The integration scheme takes advantage of the similarity between the Shibboleth and the CardSpace frameworks, and this should help to reduce the effort required for full system integration. Interoperation between CardSpace and Shibboleth may be attractive since both schemes support user authentication as well as exchange of user attributes. In addition, they both support SAML tokens. Moreover, implementation of the scheme does not require technical co-operation between Microsoft and Internet2.

Planned future work includes investigating the possibility of extending the CardSpace identity selector to simultaneously support security tokens from a variety of identity providers, such as OpenID, Liberty, Shibboleth, as well as CardSpace remote and self-issued identity providers. Possible future work may also investigate the possibility of extending the scheme to support CardSpace-enabled relying parties that employ security token services.

## ACKNOWLEDGEMENTS

The first author is sponsored by the Diwan of Royal Court, Sultanate of Oman. The helpful comments provided by anonymous referees are gratefully acknowledged.



*Security Comm. Networks* 2011; **00**:1–13 © 2011 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
*Prepared using secauth.cls*

## REFERENCES


1. Bertocci V, Serack G, Baker C. *Understanding Windows CardSpace: An Introduction to the Concepts and Challenges of Digital Identities*. Addison-Wesley, Reading, Massachusetts, 2008.
2. Cantor S, (editor). *Shibboleth Architecture — Protocols and Profiles*. Internet2 2005. `http://shibboleth.internet2.edu/shibboleth-documents.html`.
3. Fitzpatrick B, Recordon D, Bufu J, Hoyt J. *OpenID Authentication 2.0 — Final* 2007. `http://openid.net/specs/openid-authentication-2_0.html`.
4. Hammer-Lahav E, (editor). *The OAuth 1.0 Protocol*. IETF: RFC 5849 2010. `http://tools.ietf.org/html/rfc5849`.
5. Berger A. *Identity Management Systems — Introducing Yourself to the Internet*. VDM Verlag, Saarbrücken, 2008.
6. Bertino E, Takahashi K. *Identity Management: Concepts, Technologies, and Systems*. Artech House Publishers, Norwood, MA, 2011.
7. Williamson G, Yip D, Sharoni I, Spaulding K. *Identity Management: A Primer*. MC Press, Big Sandy, Texas, 2009.
8. Clippinger JH. *Higgins towards a Foundation Layer for the Social Web*. Higgins — working draft 2011. `http://www.socialphysics.org/images/Higgins6.04.06.doc`.
9. Al-Sinani HS, Mitchell CJ. Using CardSpace as a password manager. *Proceedings of IFIP IDMAN '10 — the second IFIP Conference on Policies and Research in Identity Management, November 18–19, 2010, Oslo, Norway. Volume 343 of IFIP Advances in Information and Communication Technology*, de Leeuw E, Fischer-Hübner S, Fritsch L (eds.), Springer, Boston, 18–30, 2010.
10. Mercuri M. *Beginning Information Cards and CardSpace: From Novice to Professional*. Apress, New York, 2007.
11. Brands S. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, 2000.
12. Brands S. *U-Prove Technology Overview*. Microsoft 2010.
13. Jones MB, McIntosh M, (editors). *Identity Metasystem Interoperability Version 1.0 (IMI 1.0)*. OASIS Standard 2009. `http://docs.oasis-open.org/imi/identity/v1.0/identity.html`.
14. Bajaj S, et al. *Web Services Policy Framework (WS-Policy)* 2006. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/ws-policy-2006-03-01.pdf`.
15. Della-Libera G, et al. *Web Services Security Policy Language (WS-Security Policy)* 2005. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-secpol/ws-secpol.pdf`.
16. Ballinger K, et al. *Web Services Metadata Exchange (WS-MetadataExchange)* 2006. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-mex/metadataexchange.pdf`.
17. Anderson S, et al. *Web Services Trust Language (WS-Trust)* 2005. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-trust/ws-trust.pdf`.
18. Nadalin A, Kaler C, Monzillo R, Hallam-Baker P, (editors). *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)*. OASIS Standard Specification 2006. `http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf`.
19. Microsoft Corporation and Ping Identity Corporation. *An Implementer's Guide to the Identity Selector Interoperability Profile V1.5*. 2008. `http://msdn.microsoft.com/en-us/windows/aa663320.aspx`.
20. Jones MB. *A Guide to Using the Identity Selector Interoperability Profile V1.5 within Web Applications and Browsers*. Microsoft 2008.
21. Nanda A, Jones MB. *Identity Selector Interoperability Profile V1.5*. Microsoft Corporation 2008.
22. Cantor S, (editor). *Shibboleth Architecture — Conformance Requirements*. Internet2 2005.
23. Scavo T, Cantor S, (editors). *Shibboleth Architecture — Technical Overview*. Internet2 2005. `http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf`.
24. Cantor S, Kemp J, Philpott R, Maler E, (editors). *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS 2005. `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`.
25. Scavo T, (editor). *SAML V2.0 Holder-of-Key Assertion Profile Version 1.0*. OASIS 2009. `http://www.oasis-open.org/committees/download.php/34962/sstc-saml2-holder-of-key-cd-03.pdf`.
26. Alrodhan WA. *Privacy and Practicality of Identity Management Systems*. Technical Report: RHUL–MA–2010–14 (Department of Mathematics, Royal Holloway, University of London) 2010. `http://www.ma.rhul.ac.uk/static/techrep/2010/RHUL-MA-2010-14.pdf`.
27. Dhamija R, Dusseault L. The seven flaws of identity management: Usability and security challenges. *IEEE Security and Privacy* 2008;

**6**(2):24–29.
28. Paquin C, Thompson G. *U-Prove CTP White Paper*. Microsoft 2010.
29. Paquin C. *U-Prove Technology Integration into the Identity Metasystem V1.0*. Microsoft 2010.
30. Kristol D. *HTTP State Management Mechanism*. IETF: RFC 2045 2000. `http://tools.ietf.org/html/rfc2965`.
31. Negrino T, Smith D. *JavaScript and Ajax for the Web: Visual QuickStart Guide*. 7th edn., Peachpit Press, Berkeley, CA, 2008.
32. National Institute of Standards and Technology (NIST). *Announcing the Advanced Encryption Standard (AES), FIPS 197* 2001. `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.
33. Hart J, Markantonakis K, Mayes K. Website credential storage and two-factor web authentication with a Java SIM. *Proceedings of WISTP '10 — Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices, 4th IFIP WG 11.2 International Workshop, Passau, Germany, April 12–14, 2010*, *Lecture Notes in Computer Science (LNCS)*, vol. 6033, Samarati P, Tunstall M, Posegga J, Markantonakis K, Sauveron D (eds.), Springer, Berlin, Heidelberg, 229–236, 2010.
34. Al-Sinani HS, Alrodhan WA, Mitchell CJ. CardSpace-Liberty integration for CardSpace users. *Proceedings of IDtrust '10 — the 9th Symposium on Identity and Trust on the Internet, Gaithersburg, Maryland, USA, April 13–15, 2010*, Klingenstein K, Ellison CM (eds.), ACM, New York, 12–25, 2010.
35. Al-Sinani HS, Mitchell CJ. Client-based CardSpace-OpenID interoperation. *Proceedings of ISCIS '11 — the 26th International Symposium on Computer and Information Sciences, 26–28 September 2011, London, UK*, Gelenbe E, Lent R, Sakellari G (eds.), Lecture Notes on Electrical Engineering (LNEE), Springer, London, 2011; 387–393. [Full version available at: `http://www.ma.rhul.ac.uk/techreports/2011/RHUL-MA-2011-12.pdf`].
36. Al-Sinani HS. Integrating OAuth with Information Card systems. *Proceedings of IAS '11 — the 7th International Conference on Information, Assurance, and Security, Malacca, Malaysia, 5–8 December 2011*, IEEE, 2011; 198–203. [Full version available at: `http://www.ma.rhul.ac.uk/static/techrep/2011/RHUL-MA-2011-15.pdf`].
37. Jørstad I, Van Thuan D, Jønvik T, Van Thanh D. Bridging CardSpace and Liberty Alliance with SIM authentication. *Proceedings of ICIN '07 — the 10th International Conference on Intelligence in Next Generation Networks*, Adera, Pessac, 8–13, 2007.



*Security Comm. Networks* 2011; **00**:1–13 © 2011 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
*Prepared using secauth.cls*

**Table I.** CardSpace versus Shibboleth

| | CardSpace | Shibboleth |
|---|---|---|
| Type | active client-based | redirect-based |
| IdP discovery | performed on the client (selector) | performed on the server (WAYF) |
| Phishing resistance | strong | scenario-dependent |
| Attribute exchange | supported | supported |
| Self-issued assertions | supported | unsupported |
| Identity federation | unsupported | supported |
| Proof of ownership | supported | supported |
| Pseudonyms | used (e.g. PPID) | used |
| Token format | many formats including SAML | SAML |

**Table II.** CardSpace-Shibboleth attribute mapping

| CardSpace personal cards | Shibboleth |
|---|---|
| givenname | givenName |
| surname | sn |
| givenname + surname | cn |
| emailaddress | mail |